PREFACE TO THE SECOND EDITION

There were two equal driving forces that created this second edition of USB Design By Example – the first was the advances in USB technology, not only the higher speed of USB2.0 but the many new and exciting developments with USB1.1 products. USB has broken out of its desktop-PC centric roots and is now being employed in portable devices, consumer electronics and servers. The 480Mb/s data transfer rate introduced by USB 2.0 will also extend the application range upward to include high performance video-conferencing cameras, printers and scanners. I have nothing but praise for the engineering group that defined USB 2.0 in an upwards and backwards compatible way – all existing USB 1.1 devices will continue to operate unchanged, even the same cables and connectors are used! New USB 2.0 hubs will support three data rates low, full and high speed (1.5Mb/s, 12 Mb/s and 480 Mb/s) and will allow new USB 2.0 I/O devices which can take advantage of this 40 times speed improvement. The beauty and elegance of their solution is highlighted by the **UNCHANGED** end-user model – there is **NO NEED** for the end-user to even know that there was a 1.1 spec and there is now a 2.0 spec. "New" USB 2.0 operates just the same as "old" USB 1.1. New lower-cost low-speed devices will continue to be developed which will further increase the dynamic range of USB-based I/O solutions.

My second driving force was the hundreds of e-mails I received from readers of my first book. Many of them started with "but all I want to do is..." so I have added many more examples that can be easily expanded. A recurring theme of "focus on the USB-specific aspects, assume we know how to interface a microcontroller to sensors and transducers" is addressed by adding more USB material and referring to other microcontroller design books for the "real-world" interfaces. Readers wanted more software and more explanation of that software so I have augmented the Windows material and added Linux and WindRiver real-time material. Device drivers are now covered alongside additional applications code examples. The focus of the book is still "how to add IO devices to a modern PC" and I am sure that you will have more fun! Thank you again for the many e-mails – keep sending them! I will continue to post additional examples on my web site at www.intel.com/intelpress/usb/usb.htm

Happy developing, John

(john.hyde@intel.com)

BRIEF OVERVIEW OF THIS BOOK

Chapter Descriptions

Part 1 – Essential USB Theory

Chapter 1: Adding I/O Devices to a Modern PC

In this chapter I present an overview of the USB Architecture to ensure that we are all using the same terms in the same way. I discuss the upwards and backwards compatible way that USB 2.0 was designed and I stress the continued development and use of USB 1.1 products. I explain that both I/O device design and PC host software are layered to enable easy support of additional I/O devices. The first range of the I/O devices I cover in this book are implemented as "Human Interface Devices," or HID, since the operating system already contains all of the software drivers to support this class of device. I cover all other data transfer types, with working examples, in later chapters. I also review the impact of USB on PC host design, and non-PC host design, as giving further impetus to drive the creation of even more USB-based I/O devices.

Chapter 2: Close to the wire

In this chapter I study the data transfer requests on the USB bus from the bottom up. I look at the signaling scheme on the wires and show that it reveals a packetized structure. I identify the three speeds of the bus (low, full and high) along with the mechanisms that a device uses to identify itself. I explain that sequences of packets are used to generate transactions and that three types of transactions are used during run time (interrupt, bulk, and isochronous), and control transactions are used during enumeration and run time. I mention private host-to-hub transactions for completeness but I do not cover these in detail in this I/O device design book. I then list the available control requests that a PC host can use to interact with an I/O device and a hub device. Finally, I look at a set of design tools that allows direct viewing, detection, and capture of bus packets.

Chapter 3: Getting to know you: Enumeration

In this chapter I study the reaction of a PC host to a new, unknown I/O device being added to a running system. I detail the enumeration process and note the responses expected of an I/O device. The process flow is logical, although somewhat verbose from a hardware designer's experience—all information is resident on the I/O device in data tables, called descriptors. This approach makes it easier for the PC host software to support "generic" devices and this, in turn, results in less PC host software for us to develop. I define a block diagram for a "minimum" I/O device.

Chapter 4: Choosing an I/O device

The protocol-based theory of Chapter 3 is transformed into practical implementations in Chapter 4. I describe the different methods of designing an I/O device and offer guidance on the approaches best suited for a range of applications is presented. A key decision will be the speed of the I/O device (low, full or high) and the tradeoffs and examples in this area are explained. The real world I/O is the next decision. Finally I review some of the turn-key products developed for specific interfacing tasks (such as Ethernet, ATAPI, etc).

Chapter 5: I/O Device Development Environment

In this chapter I detail the development environment options for a USB I/O device. All of the I/O solutions that I describe will require a combination of skills involving I/O device hardware, I/O device firmware, and PC host software. These are multiple and diverse skills that must be mastered. A team of three typically implements a USB I/O solution, but, with the help of the cookbook methods I describe in this book, a single developer will be able to make a lot of progress. I focus on the processes used to develop I/O device firmware in this chapter. The variety of solutions presented lets you choose the path most suitable for your project.

Chapter 6: PC Host Software Development

In this chapter I explain the USB software structure on the PC host via several example programs. The first program, a USB Device Display program, extracts and displays all of the descriptors introduced in the Chapter 3, of all of the devices currently attached via the USB bus. The program does this by searching for devices on the USB bus, so you will learn valuable techniques for the "low-level" USB activity. The second program, a HID Display program, searches for and displays information on the currently installed HID-class devices. In this program, I use a "high-level" file style of access to the I/O devices so that we can focus on the information transferred between the PC host and the I/O device, and not on the details within the USB packets.

Part 2 –USB in Practice

Chapter 7: Design example: buttons and lights

Once we get to this chapter, we have enough information to construct our first USB I/O device. I chose the simple example of writing to a remote 8-bit port and reading from a remote 8-bit port via USB so we can focus on the design **method** This example is the hardware equivalent of a "Hello World" program. I use the HID driver included with Windows so there is no OS code to write. Attaching to a HID device is a little convoluted but once we have a connection to our I/O

device then reading and writing is very easy. I work through the complete design, including hardware implementation, firmware design and implementation and PC host applications software, using the processes developed in Chapter 6.

Chapter 8: Completing the basic design

Several key design details were omitted in Chapter 7 so that more rapid progress could be made. The example design is **operational**, but it is not **compliant** with the USB Specification. I cover power management in detail in this chapter including an example of remote wakeup. A vendor ID must be obtained from the USB Implementors Forum in order to sell any I/O device as a product. I describe this registration process and the testing methods and tools that are used to ensure complaince to the USB Specification and interoperability with other devices.

Chapter 9: Expanding the basic design

The working example of Chapter 8 is extended in all possible directions – up and down! I detail the "behind-the-scenes" operation of a HID device such that an appreciation of how best to prove our I/O device can be understood. I implement multiple interfaces, multiple reports, and larger amounts of data using a variety of working examples. Some applications require the I/O device to be electrically isolated from the PC Host – I present two alternate approaches. The maximum length of a USB cable segment is 5 metres so, when using the maximum depth of hubs, the I/O device is constrained to be within 30 metres of the PC Host – I discuss the requirements for a "bus-extender" and present some alternate solutions. Finally I implement a cost-reduction exercise to tune the prototype into a production product, including a change to a different USB device.

Chapter 10: Building I/O bridges

I study the options of bridging existing standard interfaces to USB in this chapter. I employ the techniques developed in the previous chapters on the various serial and parallel interfaces found in a PC Host. The bridge to an RS-232 modem, for example, contains many redundant components, so this design example is optimized to produce a cheaper and higher-performance, USB-based modem.I then move an ISA card out of the PC to the other end of a USB cable. I then look at a variety of custom interfaces which all inherit the improved capabilities and ease-of-use of USB.

Chapter 11: Moving a lot of data

Most of the examples so far have been moving small amounts of intermittant data using a HID interface. Many applications require a large amount of data to be moved and this chapter focuses on bulk transactions. Windows does not include a simple BlockIO Class Driver so, in this chapter, I write a WDM device driver and demonstrate it with several bulk transfer examples. I also develop matching I/O device firmware for this custom interface. There is a comprehensive bulk transfer driver called the Mass Storage Class driver so I develop I/O device

firmware to interact with this class driver. I move on to present several device driver examples to show you the seemingly impossible tasks that you can accomplish in this world. These examples include device driver that can initialize a custom I/O device (LoadEz), a filter driver that Displays Usb Enumeration Transactions (DUET) and a filter driver that can modify descriptors at run time (TwoKeyboards).

Chapter 12: I like the sound of that

In this chapter we discover that handling real-time data, such as sound, is not much more difficult than handling other large amounts of data. Following an investigation of "digital audio," I have detailed the mechanism for supporting sound in USB via the full implementation of several design examples: high-quality audio output using speakers, and lower-quality, bidirectional audio using a telephone. You discover that sound is an easy attribute to add to many I/O devices.

Chapter 13: I can see you

As far as USB is concerned, digital video is just like digital audio except there is a lot more data! In this chapter I study the real amount of throughput that USB 1.1 can sustain and conclude that video data must be compressed to be usable. I develop an example that accepts video from a USB camera and displays it on the PC screen. This software is then augmented to add frame-to-frame comparisons and thus creates a "digital, motion-detector VCR".

Chapter 14: Hubs – the inside story

The hub is USB's unsung hero. From the exterior view it looks like a simple splitter/combiner but in this chapter we look inside a hub to appreciate what it adds to a USB system solution. We shall see that the hub plays a major role in USB's hot plug and play features and in its managed power distribution. Following a presentation of the different hub types I implement an example design of a compound device – an I/O device with hub features.

Chapter 15: Portable/Handheld Designs

After its initial success in the PC arena, USB is now being added to consumer devices such as portable and handheld appliances. I present a variety of client solutions including a security device, personal scanner, hard disk, an MP3 player, digital still camera and a personal data assistant (PDA).

Chapter 16: But I Don't Do Windows

All of the design examples that I have presented so far have been based on a Windows WDM platform. USB support was first integrated into Windows 98 and is now supported in Windows 2000. In this chapter I cover Linux and Windriver's VxWorks real time operating system. The "Buttons and Lights" example is implemented on both OS'es so that you can compare the solutions.

Appendix A Available OEM USB Products

This chapter is included on the CDROM has a hypertext document with live links to vendors' sites for all of the products mentioned in this book.

Appendix B Microcontroller interfacing ideas

The focus of this book is the "USB side" of the microcontroller. I learnt many useful techniques while implementing the examples and this appendix includes a variety of sensor and transducer interfacing schemes, with the matching firmware, that I used.